Imitation Learning for Locomotion and Manipulation

Nathan Ratliff
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Email: ndr@ri.cmu.edu

J. Andrew Bagnell
Robotics Institute / MLD
Carnegie Mellon University
Pittsburgh, PA 15213
Email: dbagnell@ri.cmu.edu

Siddhartha S. Srinivasa
Intel Research Pittsburgh
Pittsburgh, PA 15213
Email: siddhartha.srinivasa@intel.com

***Abstract*— Decision making in robotics often involves computing an optimal action for a given state, where the space of actions under consideration can potentially be large and state dependent. Many of these decision making problems can be naturally formalized in the multi-class classification framework, where actions are regarded as labels for states. One powerful approach to multi-class classification relies on learning a function that scores each action; action selection is done by returning the action with maximum score.**

In this work, our interest is in applying recently developed techniques for large non-linear multi-class learning to problems of imitation learning in robotics. In particular, we apply recently developed functional gradient methods for optimizing a structured margin loss function to problems in robot locomotion and manipulation. In the first case, the problem is to predict next footstep locations greedily given the four-foot configuration over a terrain height map, and the second problem is to predict good grasps of complex free-form objects given an approach direction for a robotic hand.

I. Introduction

Robot manipulation tasks usually involve a large number of actions possible at a given state. Importantly, skilled humans operators are often quite adept at choosing effective actions for a given state of the robot and can demonstrate this correct behavior. It is usually quite difficult for such an expert to articulate their strategy however; the decision is often a nonlinear combination of numerous desiderata such as stability, energy minimization, actuator limits, and future intent. It is much easier for the operator to demonstrate optimal actions than it is to carefully enumerate the complex function being optimized to produce the action. In *imitation learning*, we study algorithms that generalize from such operator demonstration to effectively chose actions for new states. Many of these learning problems can be naturally formalized in the multi-class classification framework, where actions are regarded as labels for states. [1], [2] This multi-class imitation learning approach is especially suited to robot applications because demonstration provides a natural method for an operator to specify optimality as well as to specify actions that the operator considers as "close" or equivalent (due to symmetries, for example).

The multi-class techniques we study in this paper learn a function that scores each action and returns the maximum scoring action as the optimal choice. The goal of the learning procedure is to find a scoring function that well captures the demonstrated behavior; in essence, the procedure searches for a scoring function that makes the human choices appear optimal. Recently, a framework for designing such large multi-class predictors has been developed by using functional gradient techniques to optimize simple structured-margin criteria. Importantly, this approach allows us to adapt existing, "off-the-shelf" simple regression (or binary classification) techniques to learn the potentially complicated score function, making it a modular and simple to implement technique.

Our interest in this work is the demonstration of the multi-class learning technique to solving robotic grasping problems and a quadruped locomotion task. From the machine learning viewpoint there is a surprising fundamental unity to these tasks. Both problems involve difficult to compute score functions, are relatively easy for expert operators to provide demonstrations, have a large number of actions, and can be straightforwardly optimized for the optimal action. We believe that many related robotics tasks have similar properties and may benefit from the approach taken here.

In the following, we first briefly introduce the learning technique, and the proceed to describe the experiments in detail.

II. Structured-margin techniques for multi-class classification

Many problems in imitation learning can be naturally posed as a multi-class classification problem. However, while traditional multi-class classification problems often have a relatively small number of possible class labels (typically from 3-20), multi-class classification formulations of imitation learning problems often have many class possibilities.

For instance, in our first set of experiments we take the set of classes to be the set of possible next step locations for a quadruped robot, and in the second set of experiments we take that set to be the set of preshape configurations a robot hand can take on given an approach direction. In these examples, number of class labels are 961 and 2304, respectively. In these settings, traditional margin-based classification models often fail. We next define the multi-class classification setting we use throughout this paper.

Let $\mathcal{X}$ the input domain (state space) and assume that the set of labels (actions) can potentially be different for each

Fig. 1. Locomotion testbed: Boston Dynamics LittleDog quadruped robot.

domain element $x \in \mathcal{X}$. We denote the prediction range as a set $\mathcal{Y}_x$, and the combined range as $\mathcal{Y} = \bigcup_{x \in \mathcal{X}} Y_x$. A multi-class classifier is defined by a score function $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ over these two sets. Given $x \in \mathcal{X}$, the classifier predicts the optimal scoring label

$$y^* = \arg\max_{y \in \mathcal{Y}_x} s(x, y) = \arg\max_{y \in \mathcal{Y}_x} s_x(y), \tag{1}$$

where we denote $s_x(y) = s(x, y)$ for notational convenience.

The algorithm [2] we describe here for solving large-scale multi-class classification problems is very general. In [1] we demonstrate the success of a similar technique on problems for which the number of class labels is exponential is some domain variable; in principle the set of labels can be infinite. For our purposes, the only requirement is that the optimization in Equation 1 required for making predictions can be performed efficiently. In the experiments presented in this paper, the number of labels is sufficiently small that the optimization can be implemented by a simple brute-force enumeration.

A. *Structured-margin Loss function*

Given a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, our algorithm optimizes a convex upper bound on a loss function $\mathcal{L}(x_i, y_i, y)$, which specifies the basic notion of loss on choosing label y for example x_i when the true label is y_i. For notational convenience, we often denoted the loss function as $\mathcal{L}_i(y)$ and $\mathcal{Y}_{x_i}$ as $\mathcal{Y}_i$. This upper bound is

$$r[s] = \frac{1}{N} \sum_{i=1}^{N} \left(\max_{y \in \mathcal{Y}_i} (s_i(y) + \mathcal{L}_i(y)) - s_i(y_i) \right). \tag{2}$$

If the loss function is always zero $\mathcal{L}_i = 0$, this function measures the sub-optimality of the example label. Minimizing this objective, attempts to find a score function for which the example labels are scored higher than all other labels. Choosing nonzero loss improves generalization by introducing a notion of structured-margin. Instead of requiring only that the example label is scored higher than all other labels, we require it to be better than each label y by an amount proportional to how bad we deem that label to be as measured by our loss function $\mathcal{L}_i(y)$.

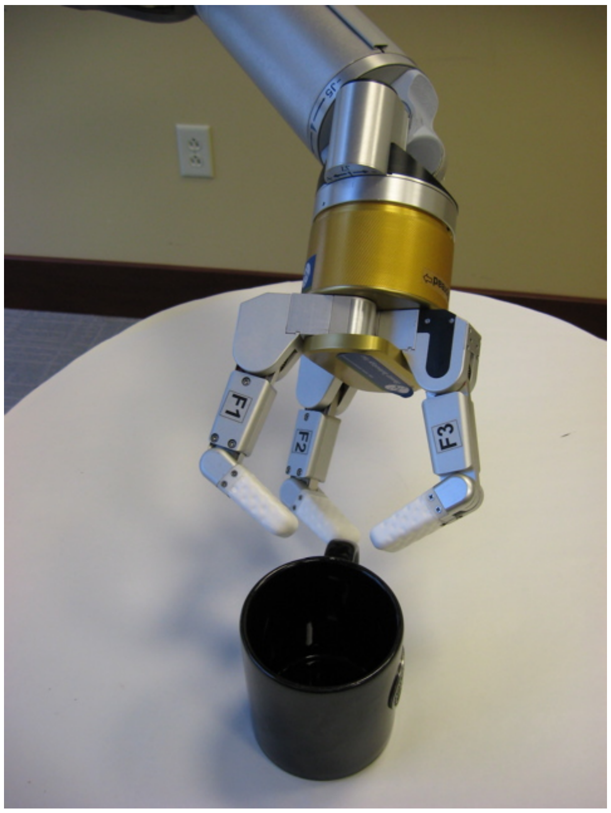

Fig. 2. Manipulation testbed: Barrett Technologies three-fingered hand.

When the number of classes is small a commonly used loss function is the binary loss. In this case,

$$\mathcal{L}_i(y) = \begin{cases} 0 & \text{when } y = y_i \\ 1 & \text{otherwise.} \end{cases}, \tag{3}$$

and reduces the margin loss of Equation 2 to the well-known SVM loss [3]. However, it is often the case in multi-class classification problems that arise in imitation learning there is a natural notion of loss, and the structured margin adapts accordingly. When the label is almost correct, i.e. has low loss, we require only to achieve a small margin over that label. Alternatively, when the label has high loss, we require a relatively large margin. See the experiments in Section **??** for examples of natural loss functions for the case when the number of classes is large.

B. *Functional gradient optimization*

In [4], a simple but effective subgradient method is developed for optimizing the upper bound 2 assuming the score function is linear in a set of features $f_i(y)$ extracted from the combined example x_i and hypothesized label $y \in \mathcal{Y}_i$. The linearity requirement is removed in [1] by generalizing the subgradient method to learning nonlinear score functions

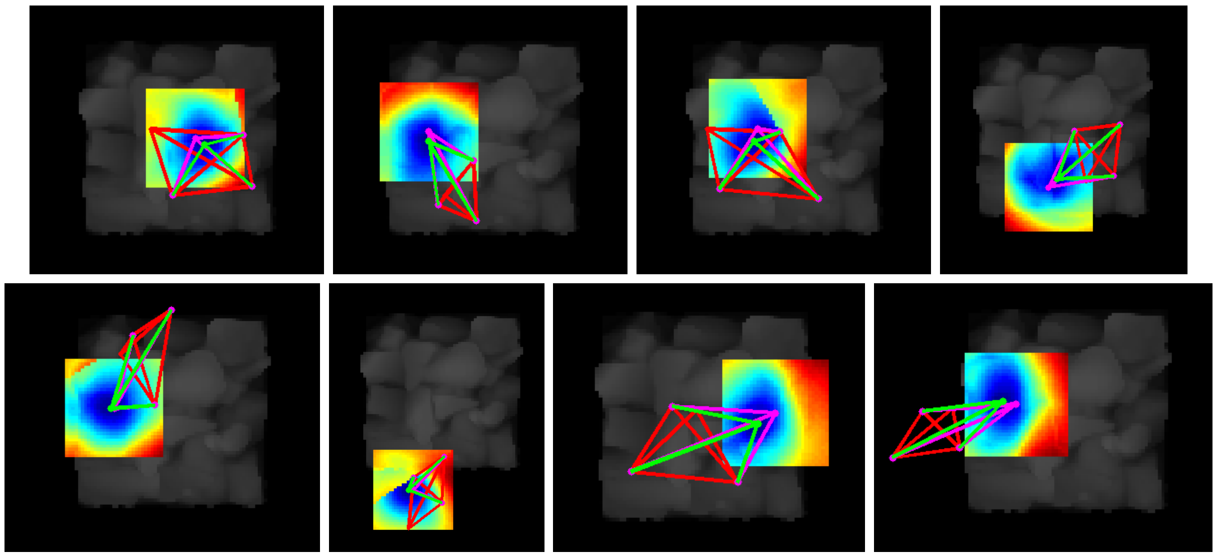
Fig. 3. Results on a number of training examples of foot placement prediction demonstrating qualitative accuracy of predictions. The original configuration is depicted as red lines connecting each of the four feet. The example step is shown in magenta, and the predicted footstep (centered at the minimum of the rendered cost function), is given in green. In the rendered cost function, bluish shades are low cost while reddish shades are high cost. In all cases, this is overlaid atop the terrain height map.

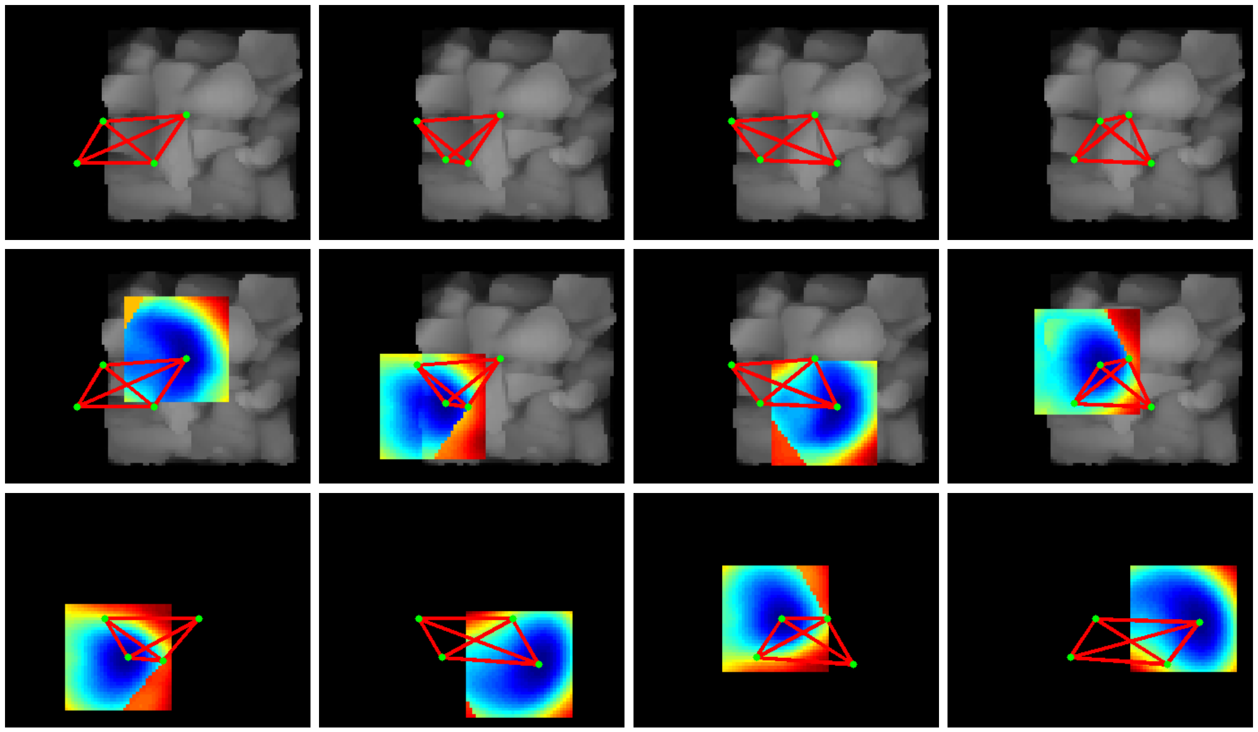
Fig. 4. Generalization of quadruped footstep placement. The four foot stance was initialized to a configuration off the left edge of the terrain facing from left to right. The images shown demonstrate a sequence of footsteps predicted by the learned greedy planner using a fixed foot ordering. Each prediction starts from result of the previous. The first row shows the footstep predictions alone; the second row overlays the corresponding cost region (the prediction is the minimizer of this cost region). The final row shows footstep predictions made over flat ground along with the corresponding cost region showing explicitly the kinematic feasibility costs that the robot has learned.

using functional gradient techniques like those first formulated in [5] and [6].

Denoting the feature vector extracted for the pair $x_i \in \mathcal{X}$ and $y \in \mathcal{Y}_i$ as $f_i(y)$, the functional gradient of the structured-margin loss(Eqn. 2) is given as

$$\nabla_s R[s] = \nabla_s \frac{1}{N}\sum_{i=1}^{N}\left(\max_{y\in\mathcal{Y}_i}(s(f_i(y)) + \mathcal{L}_i(y)) - s(f_i(y_i))\right) \tag{4}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(\delta_{f_i(y^*)} - \delta_{f_i(y_i)}\right), \tag{5}$$

where we denote the loss-augmented prediction as $y_i^* = \arg\max_{y \in \mathcal{Y}_i} s(f_i(y)) + \mathcal{L}_i(y)$. [1] A full derivation of this result, can be found in [4], however we provide here the intuition. The functional gradient is the direction in the space of score functions that would most improve performance on the loss function $r[s]$. Intuitively, the functional gradient assigns to misclassified examples an increase in score on the demonstrated action (making the classifier more likely to choose it), and a decrease in score to the action that the classifier (at the current iteration of learning) currently chooses incorrectly.

The functional gradient as defined above doesn't generalize to new states: it is tied to the particular training examples. To provide generalization to new states we rely upon the generalization ability of standard classification and regression approaches. We "project" the functional gradient onto a simpler functional form that generalizes. For instance, we will may try to find a neural network in a hypothesis space $\mathcal{H}$ of such functions that is both simple (low complexity, high prior probability) and represents the functional gradient well. Such a projection can be derived by maximizing the inner product over the hypothesis space with the functional gradient. ([4] provides a full derivation.)

$$\begin{aligned} h^* &= \arg\max_{h \in \mathcal{H}} \langle h, \; -\nabla_s R[s] \rangle & (6) \\ &= \arg\max_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \langle h, \; \delta_{f_i(y^*)} - \delta_{f_i(y_i)} \rangle & (7) \\ &= \arg\max_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} h(f_i(y^*)) - h(f_i(y_i)). & (8) \end{aligned}$$

This projection step can be implemented as a reduction to binary classification or regression using a data set generated by collecting two examples for each x_i, one corresponding to the correct label $(f_i(y_i), 1)$, and another corresponding to the current loss-augmented prediction $(f_i(y^*), -1)$. Training a binary classifier using this data set returns h^* thereby implementing the projection. Intuitively, for each i, the first example attempts to make h^* have a positive value at $f_i(y_i)$ so that adding h^* to the previous hypothesis will increase the function at that point thereby increasing the value of the correct label. Similarly, the second example attempts to make h^* negative at the current loss-augmented label, so as to reduce the score function at that point.

The technique then generalizes gradient descent: we first identify the functional gradient of the loss function, then project it using our binary classifier, and take a step in the space of score functions by adding together the resulting predictions with the previously computed "gradients".

This leads to a very simple algorithm. Given a step size sequence $\{\alpha_t\}_{t=1}^{\infty}$, the algorithm proceeds as follows:

[1]Here we've used the property that the functional gradient of a function evaluated at a point is the delta function centered at that point. In this case, $\nabla_s s(f_i(y)) = \delta_{f_i(y)}$.

```
1) initialize $s = 0$
2) for $t = 1 \ldots T$
    a) initialize $\mathcal{D} = \emptyset$
    b) for each $i = 1 \ldots N$
        i) find $y^* = \arg\max_{y \in \mathcal{Y}_i} s(f_i(y)) + \mathcal{L}_i(y)$
        ii) set $\mathcal{D} \leftarrow \mathcal{D} \cup \{(f_i(y_i), 1)\}$
        iii) set $\mathcal{D} \leftarrow \mathcal{D} \cup \{(f_i(y^*), -1)\}$
    c) train binary classifier/regressor on $\mathcal{D}$ to produce $h_t$
    d) set $s \leftarrow s + \alpha_t h_t$
3) return $s = \sum_{t=1}^{T} \alpha_t h_t$.
```

In our experiments we take the stepsize sequence to be $\alpha_t = \frac{1}{\sqrt{t}}$.

C. Exponentiated gradient variant

The algorithm above is a generalization of standard gradient descent. In many cases, like for instance when we wish the score function to be a positive number or when we add together scores over multiple states (i.e. when doing minimum cost planning instead of brute force enumeration), we instead generalize a powerful related method: exponentiated gradient descent [7], [2]. Implementing the functional version of exponentiated gradient descent is a simple modification of the algorithm above. We replace the update of the score function as a exponentiated version:

$$s \leftarrow \exp(\log(s) + \alpha_t h_t). \qquad (9)$$

Exponentiated gradient descent share similar convergence guarantees ([8]), but implements a different prior over the space of score functions. It places large prior weight on score functions with a great deal of dynamic range. We note that in this paper (unlike in [2]), exponentiating the scores doesn't change the argmax as we are not adding together scores over multiple states. Note that is does change the effect of the margin term.

III. Application

This section details experiments using the functional gradient imitation learning techniques described above on two problems: quadruped locomotion, and grasp planning, detailed in Sections III-A and III-B, respectively.

A. Quadruped Locomotion

The quadruped (Boston Dynamics' *LittleDog*) used for this experiment is depicted in Figure 1. The input state-space $\mathcal{X}$ consists of a four foot quadruped pose situated at a particular location of a 2.5-dimensional height map (i.e. a height for each x-y location), in conjunction with an "active" foot, i.e. that which is to be moved next. For a given x, the prediction range $\mathcal{Y}_x$ is the set of all possible next step locations for the action foot. In these experiments, we take this region to be a square centered at a point computed from the current four foot configuration. This region is discretized into 961 $(= 31 \times 31)$ locations.

Training examples are extracted from the intermediate poses and next step foot locations chosen by a human tele-operator remote controlling the robot across the terrain. While we could apply a full planning based solution to the imitation learning problem as in [1], a one-step look-ahead, greedy approach approach is sufficient for the terrains we considered.

Features for each possible next location fall into two categories: action features and terrain features. Action features account for the kinematic constraints of the robot as they manifest themselves in the four-foot configuration. They include the distance from the hypothesized next-step location ν and each of the original foot locations as well as the radius of the inscribed circle of the support triangle resulting from that action. Terrain features, on the other hand, contain information describing local variation in the terrain. For these experiments a very simple set of terrain features was used. Seven smoothings of the height map were generated by performing Gaussian convolutions, and the feature vector was extracted as the vector 8 responses (including the raw height) at the pixel corresponding to the desired foot location.

We apply the the above gradient boosting algorithm to this data using the following loss function

$$\mathcal{L}(\nu, \nu_i) = 1 - \exp\left(-\frac{(\nu - \nu_i)^2}{2 * \sigma^2}\right) \tag{10}$$

where ν and ν_i are the hypothesis next foot location and the example next foot location, respectively. This loss function increases rapidly from 0 at ν_i and saturates to 1 at a distance regulated by the hyperparameter σ. The space of regression algorithms we chose ($\mathcal{H}$) was small, two hidden-layer neural networks using back-propagation for training.

Figure 3 depicts a few of the training examples and the resulting predicted next step for each. Each image shows the original four-foot pose with red lines connecting each foot to emphasize the relative orientation of the original foot locations. The example and predicted next steps are depicted in magenta and green, respectively, and the learned cost map over the local search region is colored with blue shades corresponding to low cost graduating to red shades corresponding to high cost. All of this is superimposed over the terrain height map where the example resides.

The cost function learned combines the kinematic constraints of the robot with local terrain variation as can be seen in the images. Without terrain input, the cost function represents the forward stepping bias seen throughout the examples. The variation seen in the cost functions learned for each example comes from terrain components. The system learns to trade off reachability, a forward stepping bias, and local terrain considerations in a way that mimics the behavior exemplified in the data. In particular, the human had a tendency to step in local convexities (i.e. cracks) to improve walking stability and robustness. In a number of these examples, the system tends to place lower cost to such regions.

B. Grasp planning

Grasp planning is often framed as an optimization over a grasp metric evaluating the quality of a grasp configuration relative to the object being grasped. Variation in planners can be categorized into: the method used to discretize the continuous space of grasp configurations, and the grasp metric used. The discretization of the grasp space can be further segmented into a general approach direction for the hand and the configuration of the hand given the approach direction. In this work we chose to assume the approach direction has been given to us by external means for two reasons. First, [9] demonstrated that a good approach direction/point can be predicted from binocular imagery, and generalization of pinch grasping using a simple parallel jaw gripper was demonstrated for a number of previously unseen objects. Second, the true approach direction chosen for a given object is very strongly influenced by task parameters as well as environmental considerations, like workspace obstacles and the capabilities of the arm that the hand is attached to.

For this problem, a number of the features described in section III-B.2 are positively correlated with the quality of the grasp. For that reason, we chose to learn a score function rather than a cost function. Let s be the log grasp metric function in question. Then the grasp chosen for a given space of grasps $\mathcal{Y}_x$ for grasping object x is given by $y^* = \arg\max_{y \in \mathcal{Y}_x} \exp\{s(x, y)\}$.

The hand used for this experiment (Barrett technology's *Barrett Hand*) is depicted in 2. This hand has three fingers, each of which has two joints driven by a single actuator. When moving freely, the distal joint moves at a fixed rate with respect to the proximal joint, much like the motion of a human finger. One of the fingers is stationary relative to the palm, while the other two can move radially around the palm in unison, a degree of freedom which we term *fingerspread*. The three finger degrees of freedom and the fingerspread combined with the six global translation and rotation degrees of freedom gives this hand ten degrees of freedom, in total. While the two joints of the fingers are constrained, each finger has a torque redirection mechanism to transfer all force to the distal joint once the proximal link has made contact. This mechanism, called *breakaway* allows the finger to continue curling around an object even after the proximal link has made contact leading to stronger grasps.

1) Grasp demonstration: We discretize the space of grasp candidates in a way similar to that described in [10]. We define a preshape to be a configuration of the hand at a distance from the surface of the object. Given a preshape we run a simple grasp controller which moves the hand toward the object along an approach direction until it is a particular standoff distance away from the first collision with the object. At that point we close each of the fingers around the object implementing breakaway as described above.[2]

We are given an approach direction and orient the palm normal to the approach direction. This leaves us with a standoff

[2]There is an additional implementational detail which makes this controller work a little better. Since there may be occlusions (e.g. appendages in our case) that we want to avoid during the approach, we actually curl in the fingers as to their stopping point, move the hand forward until it is inside the object, open the fingers entirely, and then back the hand out of the object until it is at a particular standoff distance from the last collision point before closing the fingers around the object.

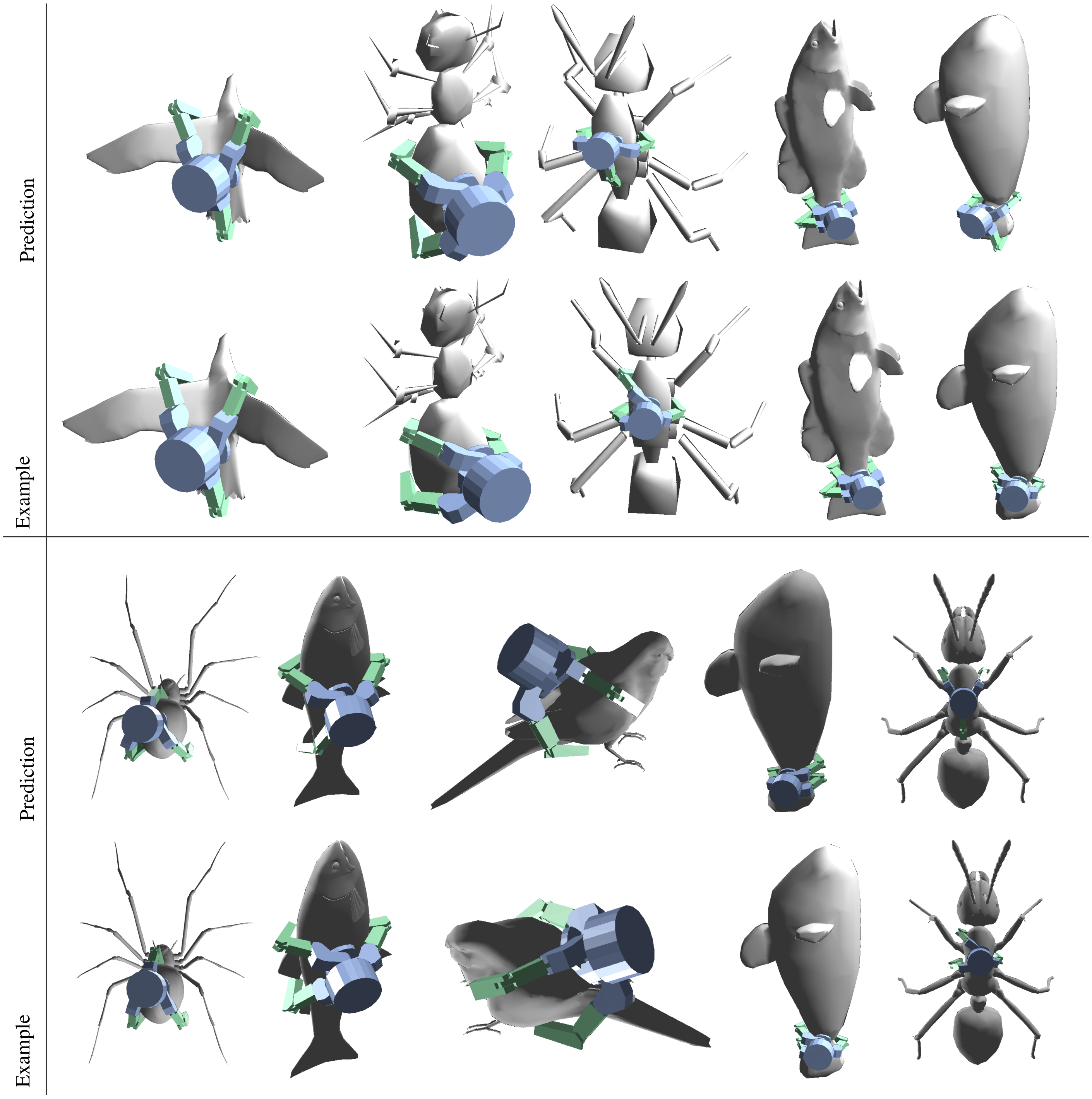

Fig. 5. Grasp prediction results on ten hold-out examples. The training set consists of 23 training examples; each test result was generated by holding the example in question out and training on the rest.

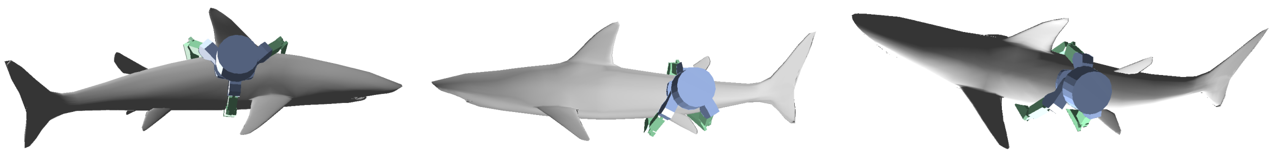

Fig. 6. Three grasps of the same object from varying approach direction.

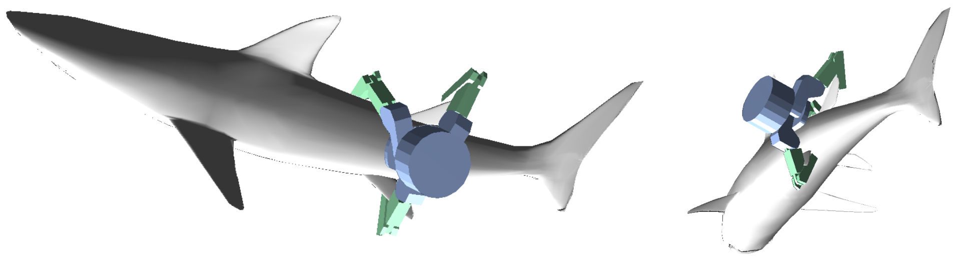

Fig. 7. A unique grasp that arises because the current feature set does not include information about fragility or flexibility of various parts of the object. Were this object perfectly rigid, this would be a reasonable grasp. The framework allows for easy addition of such extra features.

parameter as well as preshape parameters, namely the roll of the hand around the axis of approach, and the fingerspread. This gives us a three-dimensional space of grasp parameters (roll, fingerspread, standoff) which we discretize. We chose steps of size $\pi/24$ to discretize the roll and fingerspread in the ranges $(-\pi, \pi]$ and $[-\pi/24, \pi/2]$, respectively, giving 48 roll points and 13 fingerspread points. That combined with four standoff values in increments of 0.01 in the range $[0, 0.04]$ gives a total of $48 \times 13 \times 4 = 2,496$ distinct grasp parameters.

Grasps are demonstrated by a trainer by manually moving through the space of grasp parameters and selecting a good grasp. Force closure was explicitly not evaluated for these demonstrated grasps since a number of the grasps chosen by the trainer are form closure grasps that cage the object. A total of 27 training examples were generated in this way from a set of animal-like object models from the Princeton Shape Database[3] by varying approach direction and object scale. The various protrusions of the models (legs, antennae, fins, beaks) make them particularly challenging for grasping.

2) Features and loss function: To demonstrate the versatility of the learning algorithm we chose a relatively simple set of features that describe locally the shape of the object beneath each of the three fingertips and the palm, as well as the distribution of object normals in that area.

Let p and v be the point and direction of interest. We shoot a set of n rays $\mathcal{R} = \{r_i\}_{i=1}^n$ from the point p in a distribution around v and extract from the collision points $\{c_i\}_{i=1}^n$ and normals at those points $\{u_i\}_{i=1}^n$ a set of features.

The first elements of the feature vector are inversely correlated to the distance to collision: $\exp\{-\lambda\|c_i - p\|\}$. In this way the feature elements are bounded between 0 and 1, and rays that do not collide receive a value of 0.

The second set of feature elements are computed as a projection of the distribution of the vectors formed by combining the contact point with the contact normal $w_i = [c_i; u_i]$ onto the space of isotropic (diagonal covariance) Gaussians. This is simply computed by finding the vectors of means m and standard deviations s of the set $\{w_i\}_{i=1}^n$. These are then appended to the feature vector.

These ray features are computed for each finger and the palm and are then combined into a single vector representing the local relation between the hand and the object. A preprocessing step standardizes the features and then whitens them. The later is implemented by performing PCA, keeping top 10 components, and normalizing their values by the standard deviation (latent value) along the component.

[3]http://shape.cs.princeton.edu/benchmark/

These features were used primarily to show that the algorithm produces reasonable results even when using a very simple set of features. There is a large amount of information that these features do not account for which may be important in grasp prediction. Two of the most obvious of these are torque produced by the object at the grasp point (dependent on both the mass of the object and the center of mass relative to the grasp point), as well as local properties of the object such as structural integrity and surface friction. An example of where the lack of the latter piece of information comes into play is shown in figure 7. Humans have an bias toward avoiding the fins of a shark or fish when grasping because of their flexibility. However, without representing this bit of information in the feature set, the learned system utilizes the flat surfaces of the fins as though the shark were a rigid statue. Additionally, we note that a better candidate for representing the distribution of normals described above is to use wrench coordinates, which are used is computations of force volumes and force closure measures.

The loss function we used for this experiment measured the physical discrepancy between the final configurations produced by the simple controller. This is implemented as the minimum distance matching between points in the fingertips of the example configuration and corresponding points in the predicted configuration. Specifically, Set p_1, p_2, and p_3 be points in the three fingertips of the example configuration y and p'_1, p'_2, and p'_3 be corresponding points in the fingertips of the predicted configuration y'. Let Π be the set of all permutations of $1, \ldots, 3$, and denote a particular permutation as $\pi \in \Pi$ with values $\pi(i)$ for $i \in \{1, \ldots, 3\}$. We define the loss function as

$$\mathcal{L}(y, y') = \min_{\pi \in \Pi} \sum_{i=1}^{3} |p_i - p_{\pi(i)}|. \tag{11}$$

This gives low loss to configurations that are similar despite having vastly differing grasp parameters due to symmetries in the hand, while still giving high loss to configurations that are physically different.

3) Generalization: For each training example, we trained on the other 26 examples and used the final grasp metric to predict a grasp for the held out example. A single hidden layer neural network with 3 sigmoidal hidden units and a linear output is used as the weak learner ($\mathcal{H}$). Because of the variability in neural network training with random initialization, for each boosting iteration, we trained an ensemble of 10 of these base learners by simply averaging the functions resulting from 10 separate trainings of the neural network on the same data set. We ran 10 iterations of scaled conjugate gradient to train each neural network.

Figure 5 displays renderings of the resulting grasp prediction (top row) along side the example grasp the trainer would have chosen for the corresponding approach direction (bottom row). We emphasize that these are generalization results and that the system was trained without knowledge of the grasp chosen by the trainer. In particular, some of the grasp predictions are effectively the same but rotated when object symmetries make the grasps non-unique.

Occasionally, the system predicts a grasp that is not stable. This is because of the limited number of examples and a lack of task-oriented reward function. The primary goal of imitation learning in this setting, however, is to produce a grasp prediction policy that is in the neighborhood of a good policy so that a reinforcement learning algorithm can be applied effectively to directly optimize this task-oriented reward function.

Figure 6 demonstrates predicted grasps for the same object generalized to various approach directions. The prediction is fast and can be easily used bootstrap a high level planner that chooses an approach direction based on obstacles in the workspace and the kinematics of the arm.

IV. Conclusions and future work

Imitation learning in many robotic applications can be naturally posed as a large-scale multi-class classification problem. In this paper, we applied functional gradient techniques for optimizing structured-margin multi-class classification machines to two complex imitation learning problems, demonstrating their effectiveness. We are working on is making the currently brute-force search over possible actions more efficient: as the dimensionality of action spaces rises, we expect it will be necessary to consider more refined optimization procedures.

V. Acknowledgements

We would like to thank Rosen Diankov, Dmitry Berenson, Dave Ferguson, Matt Mason, and Mike Vande Weghe for their assistance and for many fruitful discussions. Nathan Ratliff was partially supported as an Intel Summer Fellow.

References

[1] N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," in *NIPS*, Vancouver, B.C., December 2006.

[2] J. A. Bagnell, J. Langford, N. Ratliff, and D. Silver, "The exponentiated functional gradient algorithm for structured prediction problems," in *The Learning Workshop*, San Juan, PR, 2007. [Online]. Available: http://snowbird.djvuzone.org/abstracts/153.pdf

[3] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, NY, USA, 1995.

[4] N. Ratliff, J. A. Bagnell, and M. Zinkevich, "(approximate) subgradient methods for structured prediction," Carnegie Mellon University, Tech. Rep., 2006. [Online]. Available: http://www.cs.cmu.edu/~ndr/mmsr

[5] L. Mason, J.Baxter, P. Bartlett, and M. Frean, "Functional gradient techniques for combining hypotheses," in *Advances in Large Margin Classifiers*. MIT Press, 1999.

[6] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," in *Annals of Statistics*, vol. 29(5), 1999a.

[7] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press, 2006.

[8] S. I. Hill and R. C. Williamson, "Convergence of exponentiated gradient algorithms," *IEEE Trans. on Signal Processing*, vol. 49, no. 6, pp. 1208–1215, 2001.

[9] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *NIPS*, 2007.

[10] A. T. Miller, S. Knoop, P. K. Allen, and H. I. Christensen, "Automatic grasp planning using shape primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.